Create a New Analysis

Introduction

This tutorial assumes that you have looked over the Analysis Framework Developer's Guide and have followed the Ergo development environment tutorial for setting up a development environment. If not, please look at those two documents because this tutorial assumes you have setup your Ergo development environment and are ready to create a new analysis plug-in so you can begin extending Ergo.

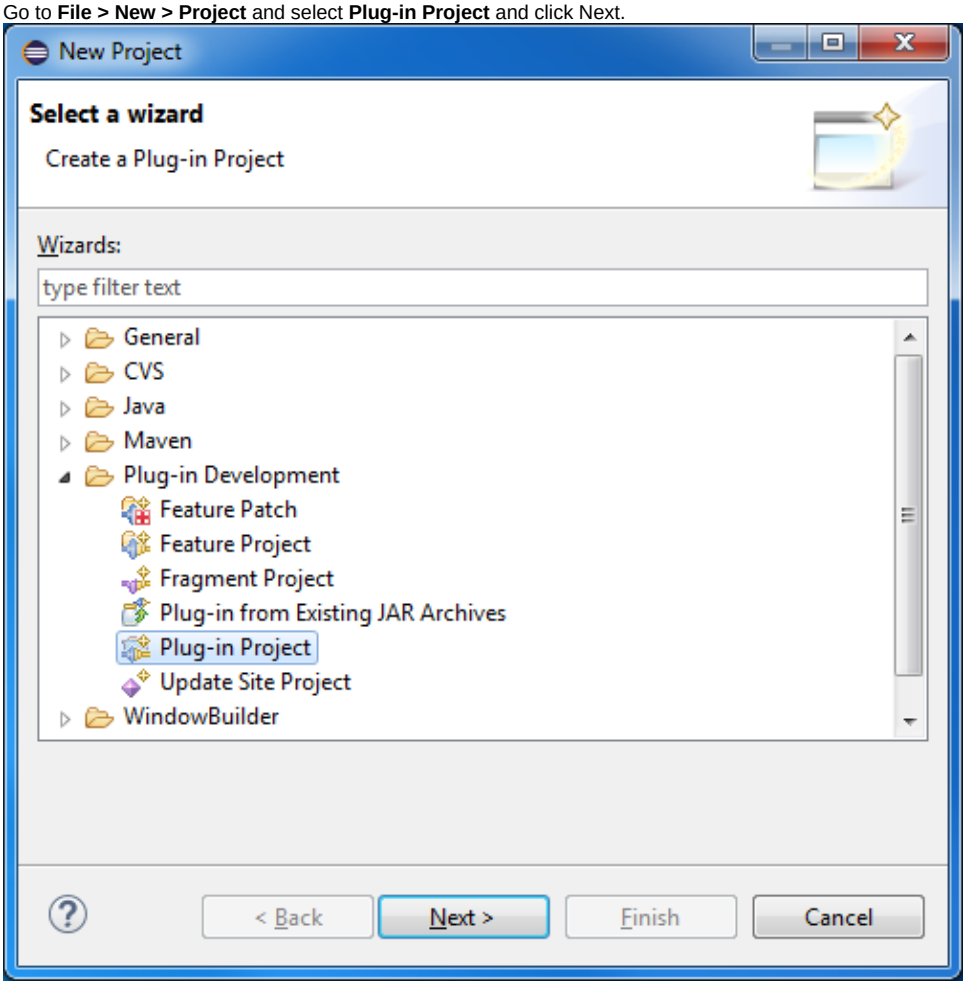
Building Analysis

The analysis we are going to perform is to check if a building is not on soft soil (0) or on soft soil (1). For simplicity, the analysis will randomly generate this number. Below you will find the steps to create this new analysis.

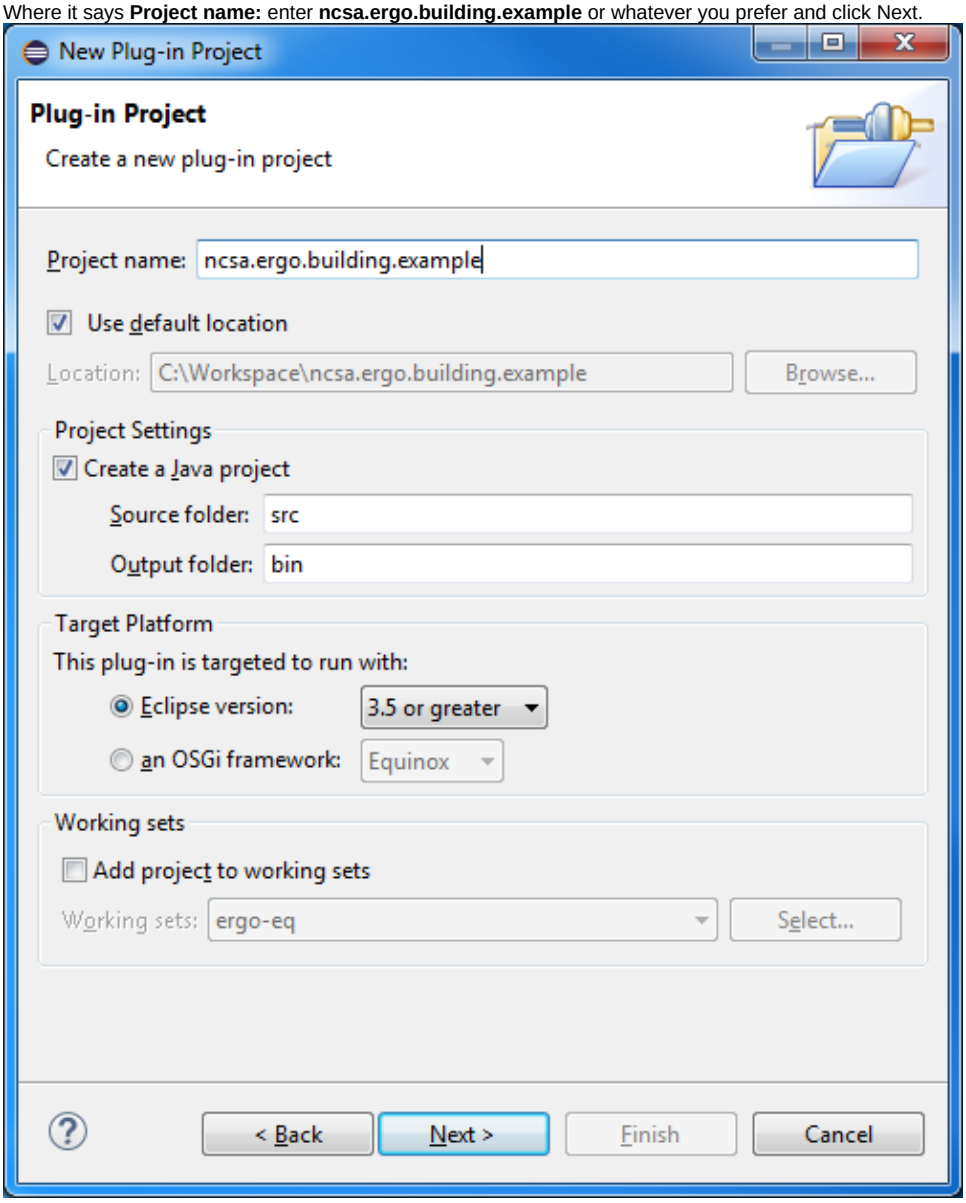
New Plug-in Project

The first step is to create a new plug-in to contain our new analysis. To do this, do the following:

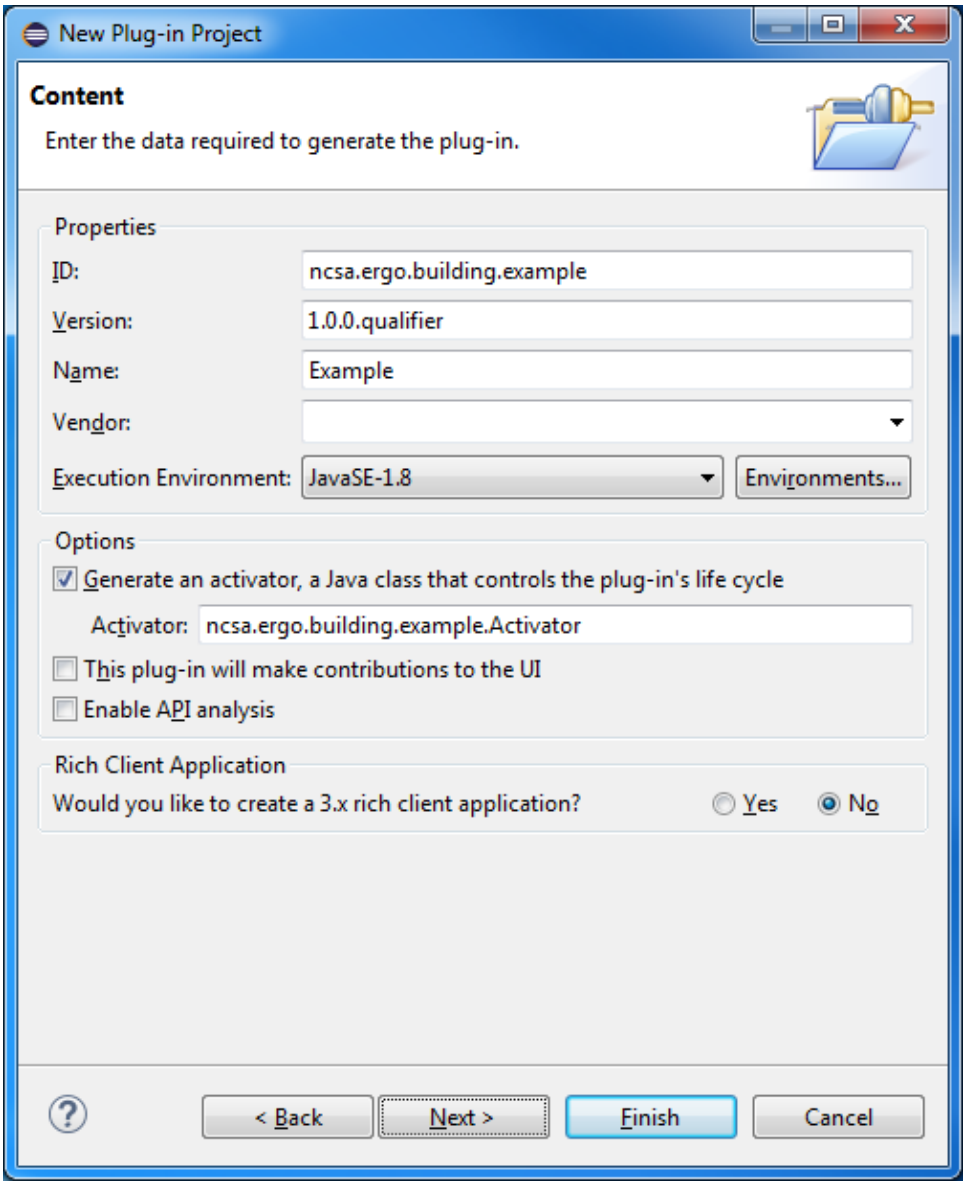
1. Go to **File > New > Project** and select **Plug-in Project** and click Next.



2. Where it says **Project name:** enter **ncsa.ergo.building.example** or whatever you prefer and click Next.



3. Nothing needs to change on this page. Make sure wizard page looks like the one below and click Finish. If the box **This plug-in will make contributions to the UI** is checked, uncheck it.



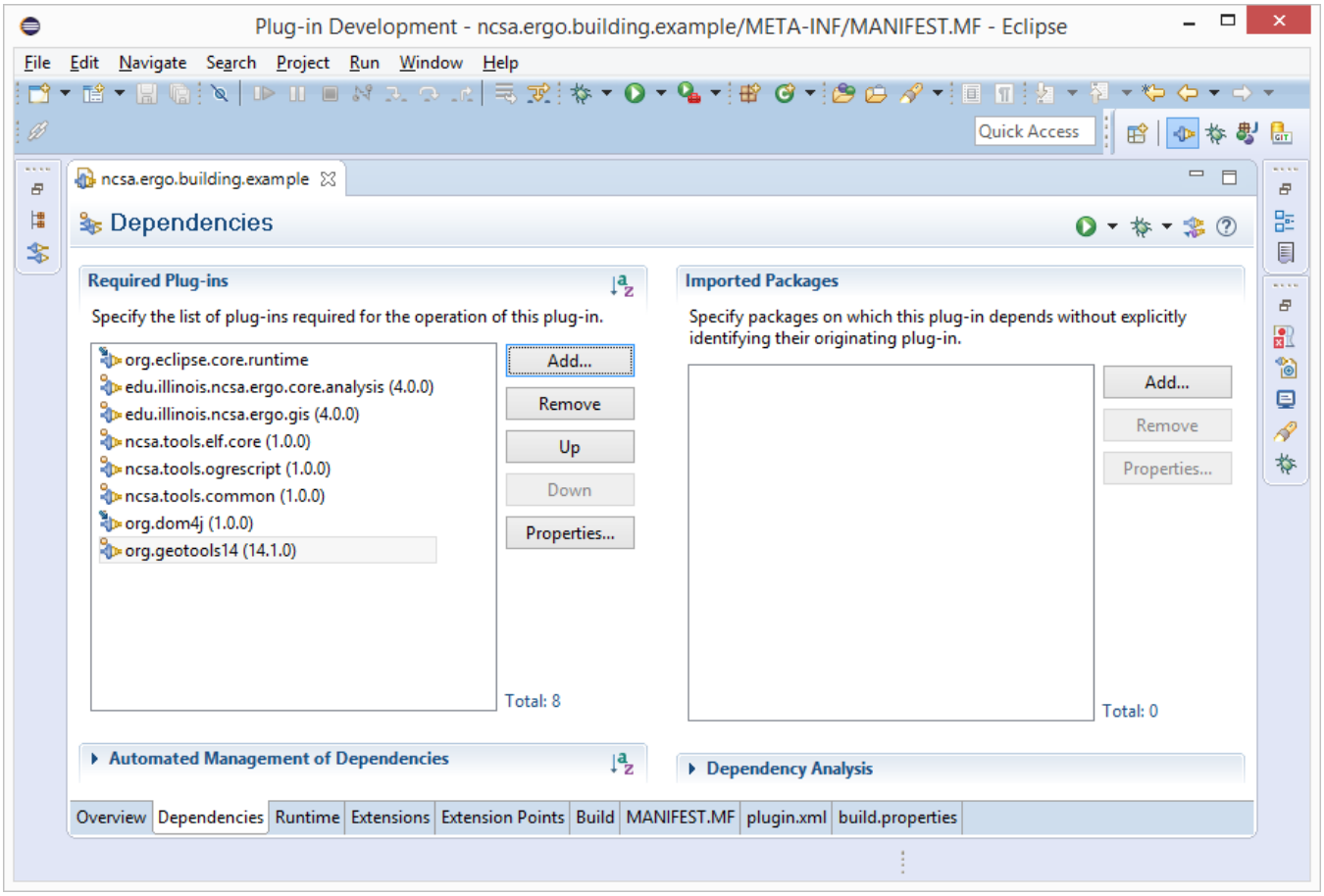
Now that we have a new plug-in, we can move on to adding the required plug-ins for our new project.

Add Required Plug-ins

We will need to add a few required plug-ins. To do this, do the following:

1. Open up the **MANIFEST.MF** file associated with our plug-in and click on the **Dependencies** tab
2. Where it says **Required Plug-ins**, click the **Add...** button and add the following plug-ins
    - edu.illinois.ncsa.ergo.core.analysis
    - edu.illinois.ncsa.ergo.gis
    - ncsa.tools.elf.core
    - ncsa.tools.ogrescript
    - ncsa.tools.common
    - org.dom4j
    - org.geotools14

Your Dependency list should look like the one in the screenshot below:



The next steps are to add a New Analysis, a Task, and a result schema that tells the analysis how to create our result.

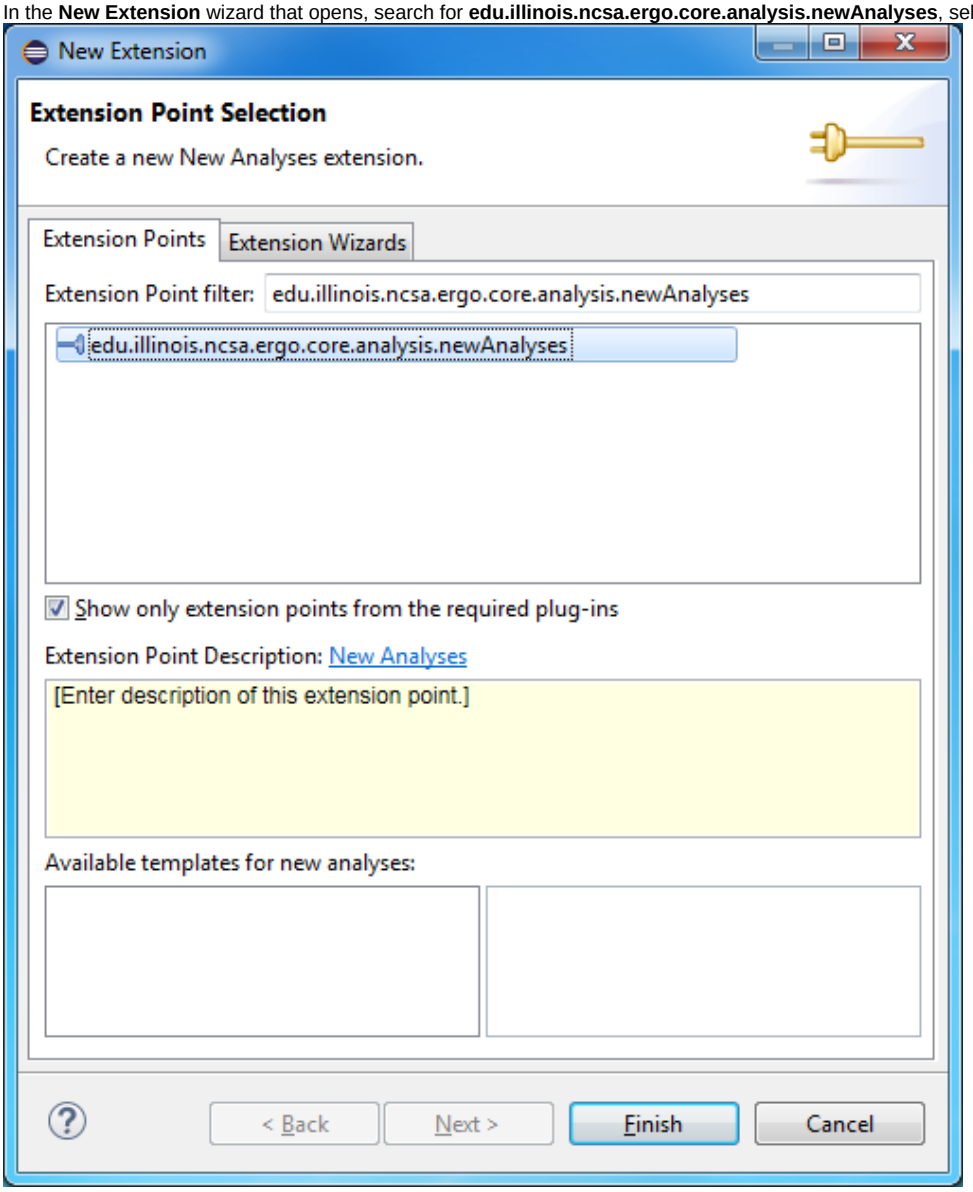
New Analysis

Analysis Description

To create a new analysis, open the **Manifest.MF** if it isn't already open and do the following

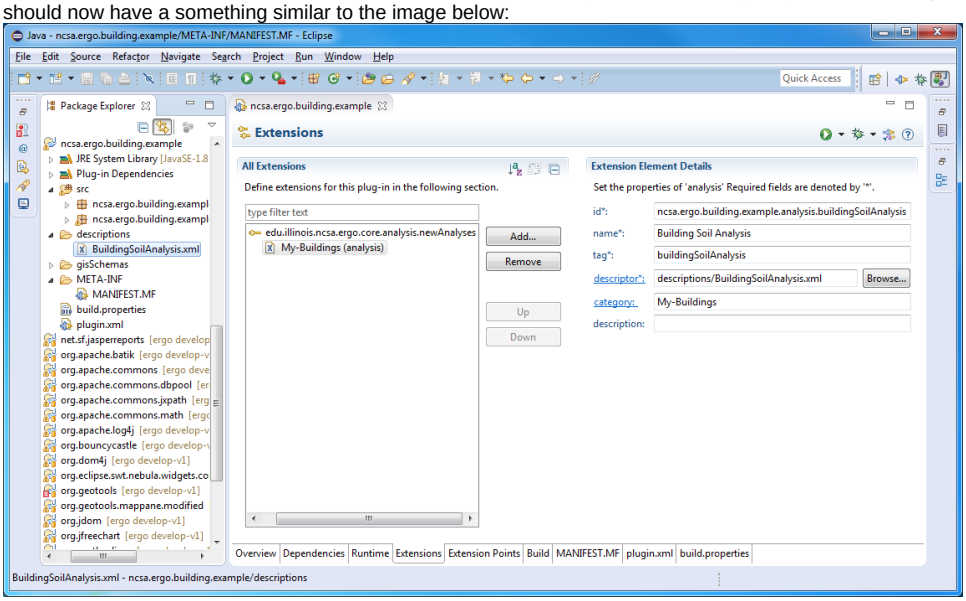
1. Click on the **Extensions** tab and click the **Add...** button

2. In the **New Extension** wizard that opens, search for **edu.illinois.ncsa.ergo.core.analysis.newAnalyses**, select it and click Finish.



3. After adding the extension point, if Eclipse did not add blank new analysis element, right click on the extension point and select **New > analysis**
4. The next step is to fill in the required parts for the new analysis. Enter the following information
    - **id: ncsa.ergo.building.example.analysis.buildingSoilAnalysis**
    - **name: Building Soil Analysis**
    - **tag: buildingSoilAnalysis**
    - **category: My-Buildings**

5. The final requirement is a descriptor. I recommend creating a new folder inside your plug-in called descriptions and adding a new file called **BuildingSoilAnalysis.xml** that we will fill in later. After creating this file, you will need to specify it in the **descriptor** field of your new analysis. You should now have a something similar to the image below:



Your **BuildingSoilAnalysis.xml** file should contain the following xml statements:

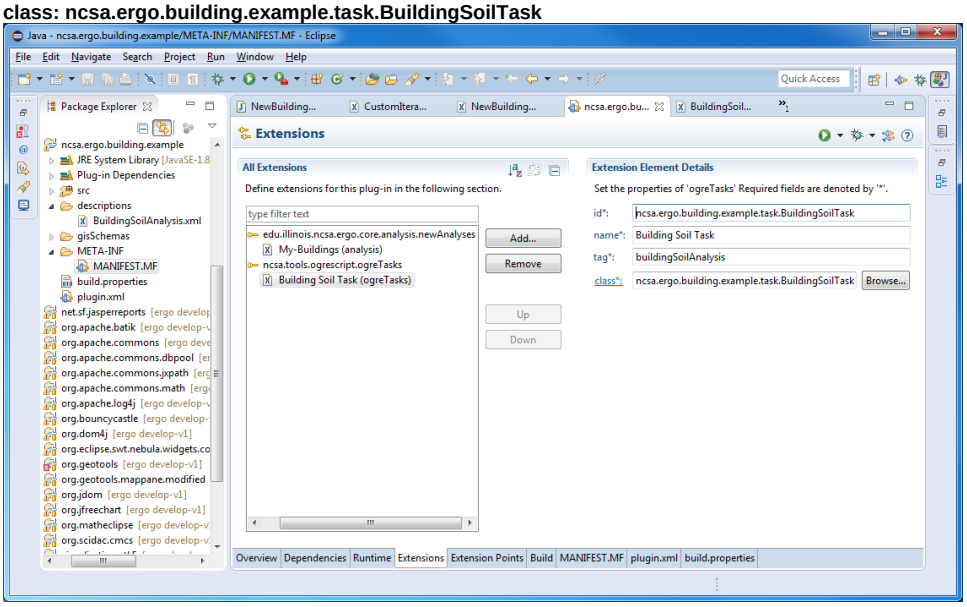
```
<analysis-description id ="ncsa.ergo.building.example.analysis.buildingSoilAnalysis">
    <analysis-type type="simpleIteration">
    <property name="iteratingDatasetKey" value="buildings">
    </property>
    </analysis-type>
    <groups>
        <group-name>Required</group-name>
    <group-name>Advanced</group-name>
    </groups>

    <!-- Analysis Inputs -->
    <!-- the result name must be prefixed with the tag of the analysis, in this case buildingSoilAnalysis -->
    <parameter format="resultName" phylum="string" cardinality="single" key="buildingSoilAnalysis.resultName" frien
dly-name="Result Name"/>

    <parameter phylum="dataset" cardinality="single" key="buildings" friendly-name="Buildings">
    <types>
        <type>buildingv4</type>
        <type>buildingv5</type>
    </types>
    </parameter>

    <!-- Analysis Outputs -->
    <output friendly-name="Building Soil Results" key="buildingSoilAnalysis" phylum="dataset" format="shapefile" ge
om="buildings" guids="buildings">
        <property name="buildings" type="base-dataset-key" value="buildings"/>
    <property name="schema" type="schema" value="ncsa.ergo.building.example.schemas.buildingSoilResults.v1.0"/>
    </output>
</analysis-description>
```

Analysis Task

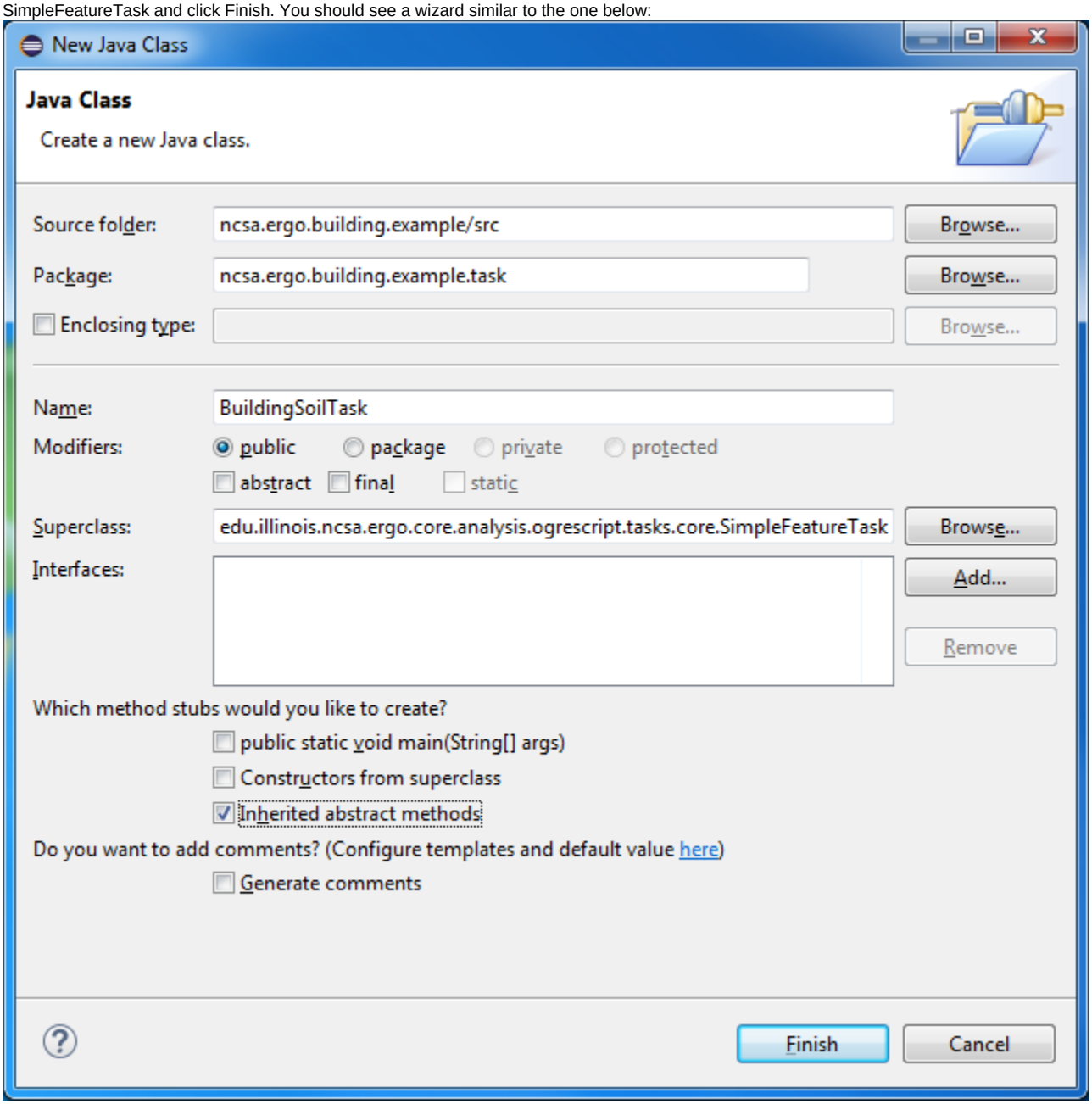
The next step is to create a new analysis task. This is the Java code that will be executed by the new analysis to produce our building soil analysis output. To create a new task, do the following:

1. Click on the **Extensions** tab and click the **Add...** button
2. In the **New Extension** wizard that opens, search for **ncsa.tools.ogrescript.ogreTasks**, select it and click Finish.
3. After adding the extension point, if it did not add a blank task element, right click on the extension point and select **New > ogreTasks**.
4. For the new analysis task, enter the following details
    - **id: ncsa.ergo.building.example.task.BuildingSoilTask**
    - **name: Building Soil Task**
    - **tag: buildingSoilAnalysis**

- **class: ncsa.ergo.building.example.task.BuildingSoilTask**



It is critical that the **tag** for the Task be identical to the **tag** for the analysis since this is how Ergo determines which task goes with which analysis. After entering the text for the **class** field, if you click on the **class**, a source file will be generated for you. Where it says **Superclass** locate the class called SimpleFeatureTask and click Finish. You should see a wizard similar to the one below:



Now that you have your class file, add the missing lines of code from the example below:

BuildingSoilTask extends SimpleFeatureTask

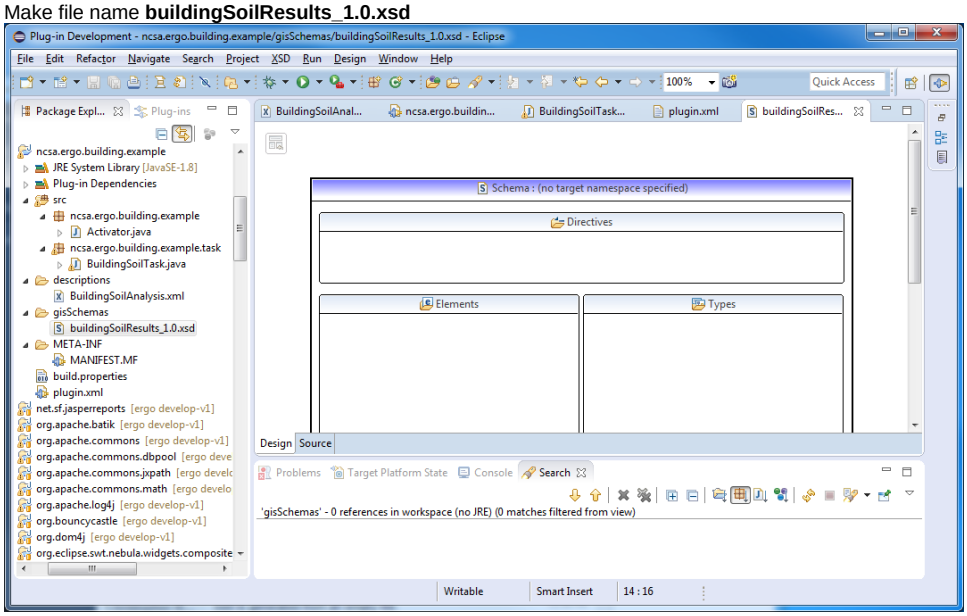
```
@Override
protected void handleFeature( IProgressMonitor monitor ) throws ScriptExecutionException{
  resultMap.put( "soiltype", isSoftSoil() );
}

/**
     * Generate a number between 0 and 5 (exclusive)
     * 0 - Soil Type A
     * 1 - Soil Type B
     * 2 - Soil Type C
     * 3 - Soil Type D
     * 4 - Soil Type E
     * @return Soil Type
```

```
     */
private int isSoftSoil() {
  Random rand = new Random();
  return rand.nextInt( 5 );
}
```

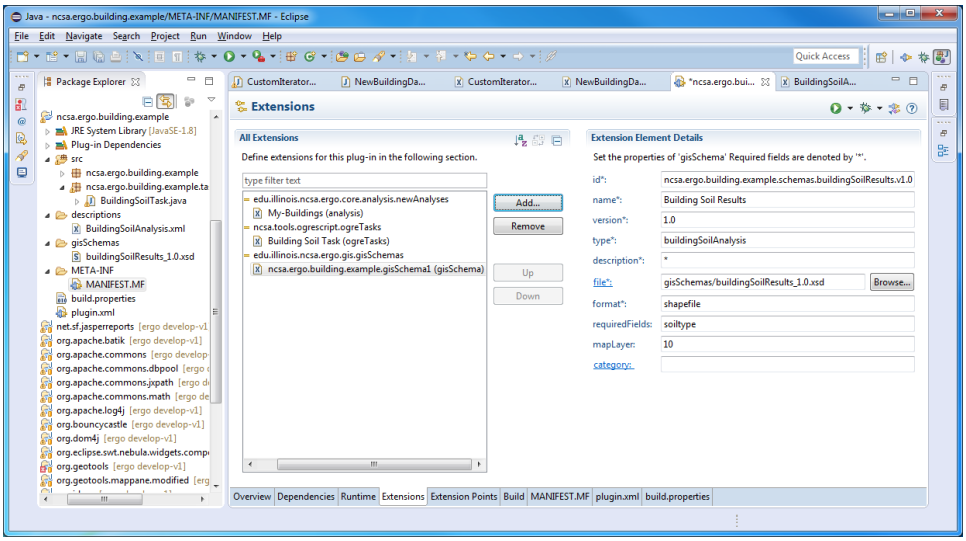
Analysis Result Type

The next step is to create a result type schema to specify the new fields created by our analysis. This tutorial will not go into detail about the schema file since it is behind the scope of this tutorial.

1. Click on the **Extensions** tab again and click the **Add...** button
2. In the **New Extension** wizard that opens, search for **edu.illinois.ncsa.ergo.gis.gisSchemas**, select it and click Finish.
3. After adding the extension point, if it did not add a blank schema element, right click on the extension point and select **New > gisSchema**.
4. Create the **gisSchemas/buildingSoilResults_1.0.xsd** file under **ncsa.ergo.building.example**.
5. Right click on the ncsa.ergo.building.example in package explorer and select **New>Folder**.
6. Make folder name **gisSchemas** and click Finish.
7. Right click on the gisSchemas folder that just created select **New > File**
8. Make file name **buildingSoilResults_1.0.xsd**



9. For the new result type, enter the following details
    - **id: ncsa.ergo.building.example.schemas.buildingSoilResults.v1.0**
    - **name: Building Soil Results**
    - **version: 1.0**
    - **type: buildingSoilAnalysis**
    - *description: *
    - **file: gisSchemas/buildingSoilResults_1.0.xsd**
    - **format: shapefile**
    - **requiredFields: soiltype**
    - **mapLayer: 10**



For the schema file **buildingSoilResults_1.0.xsd**, please enter the following information:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:gml="http://www.opengis.net/gml" xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="h
ttp://www.ionicsoft.com/wfs" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:iwfs="http://www.ionicsoft.com/wfs"
targetNamespaceOptionnal="true" xmlns="http://www.ionicsoft.com/wfs" elementFormDefault="qualified">
    <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/2.1.2
/feature.xsd"/>
    <xsd:element name="building-soil-analysis" substitutionGroup="gml:_Feature" type="iwfs:building-soil-analysis"
/>
    <xsd:complexType name="building-soil-analysis">
      <xsd:complexContent>
        <xsd:extension base="gml:AbstractFeatureType">
          <xsd:sequence>
            <xsd:element name="ergo.soiltype" minOccurs="0" nillable="true" type="xsd:integer"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:schema>
```

Example Plug-in

If you have setup your Ergo development environment and have connected to our Git repository, then you can find this example plug-in by going to the repository at importing the project with the name ncsa.ergo.eq.building.example. This will checkout the plug-in to your workspace.

ssh://git@opensource.ncsa.illinois.edu:7999/ergo/tutorial.git (use your own id and password instead of git)

https://opensource.ncsa.illinois.edu/stash/scm/ergo/tutorial.git